1 ***(1 punto)*** *Enumere los ciclos de bus que se producen durante una secuencia de reconocimiento de interrupciones con vectorización en la que la dirección de comienzo de la rutina de tratamiento de interrupción se obtiene mediante direccionamiento relativo al registro base de la tabla de vectores.*

→ Para salvaguarda Reg. Estado y PC : 2
→ Leer tabla y obtiene el específico: 2.

SOLUCIÓN

2 ***(1 punto)*** *Desde el punto de vista de* **duración total** *de las operaciones de E/S cuantifique el impacto de las diferentes técnicas para la realización de las operaciones de E/S.*

↳ afectan + a la CPU q al periférico.

SOLUCIÓN

3 *Sea un computador con una CPU de 64 bits, capaz de ejecutar 2.000 MIPS y cuya secuencia de reconocimiento de interrupciones (SRI) tiene una duración equivalente a 10 instrucciones.*

Este computador tiene conectada una unidad de disco duro con las siguientes características:

- *Velocidad de transferencia:* 10^8 *bytes/s.*
- *Tiempo medio de acceso: 4 ms.*
- *Registro de datos de 64 bits.*
- *Tamaño del sector: 1.024 bytes.*

El módulo de E/S de la unidad de disco opera mediante interrupciones con las siguientes características:

- *Las rutinas de inicio y fin de una operación de E/S constan de 100 y de 50 instrucciones respectivamente.*
- *Su rutina de tratamiento de las interrupciones consta de 40 instrucciones.*

a) (1 punto) *Calcule cuántos de estos discos duros podrían operar simultáneamente.*

b) (2 puntos) *Calcule el tiempo total de una operación de lectura de un sector del disco duro así como el tiempo de CPU que se emplea.*

Como se necesita que 8 discos duros puedan operar simultáneamente, se conectan sus módulos de E/S para operar por DMA. Considere que las rutinas de inicio y fin de una operación de E/S constan igualmente de 100 y de 50 instrucciones respectivamente.

c) (1 punto) *Calcule cuántos de estos discos duros podrían operar simultáneamente si el protocolo de concesión y liberación de los buses dura 2 ns en total y el ciclo de acceso a memoria 2 ns.*

d) (1 punto) *Calcule el tiempo de CPU que se emplea ahora en una operación de lectura de un sector del disco duro.*

Este computador dispone también de una conexión a una red Ethernet con las siguientes características:

- *Velocidad de transmisión de 1 Gigabit por segundo (*10^9 *bits/s).*
- *Buffer de 4 registros de datos de 64 bits.*
- *Las rutinas de inicio y fin de una operación de E/S para la transferencia de un bloque de datos constan de 150 y de 90 instrucciones respectivamente.*
- *Operación por interrupciones.*

e) (1 punto) *Calcule el número máximo de instrucciones que puede tener la rutina de tratamiento de interrupción para que este dispositivo pueda operar simultáneamente con los 8 discos duros.*

Considere a partir de aquí que la rutina de tratamiento de interrupción ejecuta un total de 110 instrucciones.

f) (1 punto) *Calcule el tiempo de CPU que se emplea en la transmisión de un paquete de 1.024 bytes.*

g) (1 punto) *Calcule el porcentaje de utilización de CPU que se puede alcanzar cuando operan simultáneamente 4 discos duros y este dispositivo.*

SOLUCIÓN

a) Se calcula la capacidad de procesamiento necesaria para atender las interrupciones de un disco duro.

$$\begin{aligned} CP_{HD} &= Freq_{int} \times I_{int} = \frac{10^8\ Byte/s \cdot 8\ bits/Byte}{64\ bits} \times (10\ instr + 40\ instr) = \\ &= 12,5 \cdot 10^6\ Hz \times 50\ instr = 625 \cdot 10^6\ instr/s = 625\ MIPS \end{aligned}$$

Se podrían atender a un máximo de 2.000 MIPS/625 MIPS = 3 discos duros

b) Se calcula el tiempo total y el de CPU de una operación del disco teniendo en cuenta que a 2.000 MIPS una instrucción se procesa en 0,5 ns.

$$\begin{aligned} t_{op} &= t_{ini} + t_{acc} + t_{trans} + t_{última-interrupción} = t_{ini} + t_{acc} + t_{trans} + (t_{sri} + t_{int} + t_{fin}) = \\ &= 50\ ns + 4 \cdot 10^6\ ns + \frac{1.024\ bytes}{10^8\ bytes/s} + (5\ ns + 20\ ns + 25\ ns) = 4.010.340\ ns \\ t_{cpu} &= t_{ini} + N_{interrupciones} \times t_{int} + t_{fin} = \\ &= 50\ ns + \frac{1.024\ bytes}{8\ bytes} \times (5\ ns + 20\ ns) + 25\ ns = 3.275\ ns \end{aligned}$$

c) Se calcula la capacidad de procesamiento necesaria para atender los robos de ciclo de un disco.

$$\begin{aligned} CP_{HD} &= Freq_{robos} \times I_{robo} = 12,5 \cdot 10^6\ Hz \times ((2\ ns + 2\ ns) \times 2 \cdot 10^9\ instr/s) = \\ &= 12,5 \cdot 10^6 \times 8\ instr/s = 100\ MIPS \end{aligned}$$

Luego pueden operar simultáneamente 2.000 MIPS/100 MIPS = 20 discos duros.

d) Se calcula el tiempo de CPU ocupada en una operación del disco operando mediante DMA.

$$t_{cpu} = t_{ini} + N_{robo-ciclo} \times t_{robo-ciclo} + t_{int} = 50\ ns + \frac{1.024\ bytes}{8\ bytes} \times 4\ ns + (5\ ns + 25\ ns) = 592\ ns$$

e) Se calcula la capacidad de procesamiento necesaria para atender las interrupciones de la unidad de red que debe ser menor o igual a 1.200 MIPS ya que 8 discos duros necesitan una capacidad de procesamiento de 800 MIPS.

$$\begin{aligned} CP_{Ethernet} = Freq_{int} \times I_{int} = \frac{10^9\ bits/s}{4 \cdot 64\ bits} \times (10\ instr + X\ instr) &\leq 1.200\ MIPS \\ 3,90625 \cdot 10^6 \times (10 + X)\ instr/s = 39,0625 \cdot 10^6\ instr/s + 3,90625 \cdot 10^6 \cdot X\ instr/s &\leq 1.200\ MIPS \\ 3,90625 \cdot 10^6 \cdot X\ instr/s &\leq 1.160,9375\ MIPS \\ X &\leq 297,2\ instr \end{aligned}$$

El número máximo de instrucciones es 297.

f) Se calcula el tiempo de CPU de una transmisión de un paquete de 1.024 Bytes por la red Ethernet.

$$\begin{aligned} t_{cpu} &= t_{ini} + N_{interrupciones} \times t_{int} + t_{fin} = \\ &= 75\ ns + \frac{1.024\ bytes}{4 \cdot 8\ bytes} \times (5\ ns + 55\ ns) + 45\ ns = 2.040\ ns \end{aligned}$$

g) Se calcula la capacidad de procesamiento necesaria para atender las interrupciones de la unidad de red.

$$CP_{Ethernet} = Freq_{int} \times I_{int} = \frac{10^9\ bits/s}{4 \cdot 64\ bits} \times (10\ instr + 110\ instr) = 468,75\ MIPS$$

Luego la capacidad de procesamiento necesaria para atender las interrupciones de la unidad de red y de cuatro discos duros es 468.75 MIPS + 400 MIPS = 868,75 MIPS que supone el 43,43 % de los 2.000 MIPS que ejecuta la CPU.

1 ***(2 puntos)*** *Indique y justifique si es verdadera o falsa cada una de las siguientes afirmaciones sobre el sistema de memoria:*

a) Las políticas de ubicación asocian a cada dirección del mapa de memoria principal una dirección del mapa de memoria caché.

b) La política de ubicación más conveniente para organizar una TLB es la directa.

*c) Las memorias caché de instrucciones utilizan habitualmente escritura inmediata (*write-trough*), mientras que las caché de datos suelen utilizar escritura aplazada (*copy-back*).*

d) La traducción de direcciones virtuales a direcciones físicas consume un tiempo que sería prohibitivo si no se utilizase una caché específica para hacer las traducciones.

SOLUCIÓN

a) Falso. La memoria caché no tiene un mapa de memoria, se accede a ella usando las direcciones de memoria principal.

b) Falso. La más conveniente sería una totalmente asociativa.

c) Falso. Las memorias caché de instrucciones no tienen definida una política de escritura, porque en ellas no se escribe. Las de datos pueden utilizar escritura inmediata o aplazada.

d) Verdadero. La caché específica es la denominada TLB.

2 ***(1 punto)*** *Describa en qué consiste el fenómeno de proximidad de referencias (*locality*). Indique qué es la proximidad espacial y temporal, y explique, razonadamente, si se observa en accesos a código y en accesos a datos. ¿En qué medida afecta este fenómeno al adecuado funcionamiento de la jerarquía de memoria?*

SOLUCIÓN

Se denomina "proximidad de referencias (*locality*, en inglés)" al hecho de que los programas accedan repetidamente a las mismas direcciones o a direcciones cercanas. Se distingue entre proximidad temporal –la traza del programa repite las mismas direcciones en instantes cercanos en el tiempo– y espacial –cuando la traza contiene direcciones próximas en el espacio de direcciones–.

La proximidad temporal tiende a darse en el código, principalmente por la existencia de bucles (o saltos "hacia atrás"), que vuelven a hacer referencia a las mismas direcciones de instrucciones, aunque también las correspondientes datos a las que este código hace referencia, o a posiciones cercanas. El hecho de que las direcciones de código tiendan a ser contiguas a partir de una bifurcación justifica la ocurrencia de la proximidad espacial para instrucciones, mientras que la de datos se da en la manera en que los compiladores almacenan las estructuras de información, en particular las de tipo vector o formación, aunque también lo suelen hacer en el caso de otras con apariencia menos "regular", como las estructuras dinámicas o los espacios de almacenamiento temporal durante la ejecución, como puedan ser la pila y el *heap* o "montón".

Precisamente el funcionamiento de la Jerarquía de Memoria (JM) se basa en la existencia de la proximidad de referencias, ya que los dispositivos más rápidos necesariamente tienen una capacidad menor, y contienen siempre la información más reciente o "fresca" y que volverá a ser referenciada. La JM no funcionaría si no se diese este fenómeno.

3 **(7 puntos)** *Sea un computador con memoria virtual y con datos y direcciones físicas de 32 bits, que tiene las siguientes características:*

- *Direcciones virtuales de 44 bits, con dos niveles de tablas de páginas, 2^{14} entradas en el primer nivel de tablas, y páginas de 64KB. Además dispone de dos TLB asociativas, una para instrucciones (TLBi) y otra para datos (TLBd), ambas con 4 entradas.*
- *Memoria caché de instrucciones (MCaI): 128 KB, directa, con bloques de 32 bytes. Política de lectura* out of order fetch.
- *Memoria caché de datos (MCaD): 256 KB, asociativa por conjuntos, conjuntos de 8 bloques, cada bloque de 32 bytes, con política de escritura inmediata sin actualización (WTWNA). Política de lectura* out of

order fetch.

En este computador, con las memorias caché y las TLBs inicialmente vacías, se ejecuta el siguiente fragmento de código, situado en la dirección 0:

```
(1)        add r10,  r0, 64       ; fin        fin = 64;
(2)        add r11,  r0, 0x10000  ; a[]       i = 0;
(3)        add r12,  r0, 0x14000  ; b[]       while (i < fin) {
(4)        add r13,  r0, 0x20000  ; sum[]         sum[i] = a[i] + b[i];
(5)        add r14,  r0, 0x24000  ; dif[]         dif[i] = a[i] - b[i];
(6)        add r20,  r0, r0       ; i             i++;
(7)loop:   ld   r1, r11, r20      ;            }
(8)        ld   r2, r12, r20
(9)        add  r3,  r1, r2
(10)       st   r3, r13, r20
(11)       sub  r4,  r1, r2
(12)       st   r4, r14, r20
(13)       add r20, r20, 4
(14)       sub r10, r10, 1
(15)       bnz  r10, $loop
```

Sabiendo que cada instrucción y cada dato ocupan una palabra determine:

a) El formato de las direcciones virtuales.

b) El formato de las direcciones físicas tal como se interpreta por el controlador de cada una de las memorias caché.

c) Justifique si es posible solapar la traducción de direcciones con el acceso a las correspondientes memorias caché. En caso negativo, razone cuál será el tamaño de página necesario para poder solapar el acceso a las TLB's con el de las memorias caché.

d) La tasa de aciertos de cada TLB.

e) La tasa de aciertos de cada memoria caché.

f) Suponiendo que las TLBs y las memorias caché están inicialmente vacías y sabiendo que el tiempo de acceso de las TLB es de 2 ns, el de las memorias caché de 3 ns y el de la memoria principal 60 ns, calcule el tiempo invertido en los accesos a datos realizados por el fragmento de código anterior. Calcule asimismo el tiempo medio invertido en los accesos a datos.

SOLUCIÓN

a) Al tener dos niveles de tablas de páginas, la dirección virtual se interpretará formada por un identificador de la entrada de la tabla de páginas de primer nivel, otro de la correspondiente de segundo nivel y un desplazamiento dentro de la página, este último de 16 bits, ya que se utilizan páginas de $2^{16} bytes$. Las tablas de páginas de primer nivel contienen 2^{14} entradas, según especifica el enunciado, por lo que se requieren 14 bits para identificar cada entrada. Finalmente, los $44 - 14 - 16 = 14$ *bits* restantes identificarán una entrada en la tabla de segundo nivel:

TP1	TP2	desplazamiento
14 bits	14 bits	16 bits

b) El tamaño de la memoria caché de instrucciones es de 128 KB, es decir, 2^{17} bytes. Al tener organización directa, su controlador interpretará una dirección física (de 32 bits según el enunciado) como formada por los campos *etiqueta*, *bloque* y *byte*. Siendo los bloques de 2^5 bytes (32 bytes), habrá un total de $2^{17}/2^5 = 2^{12} = 4096$ bloques, por lo que la interpretación será la siguiente:

etiqueta	bloque	byte
15 bits	12 bits	5 bits

La caché de datos tiene un tamaño de 256 KB, es decir, 2^{18} bytes. Al tener organización asociativa por conjuntos, su controlador interpretará que una dirección física de 32 bits está formada por los campos *etiqueta*, *conjunto* y *byte*. Siendo los bloques de 2^5 bytes (32 bytes), habrá un total de $2^{18}/2^5 = 2^{13} = 8192$ bloques y, puesto que los conjuntos son de 8 bloques, habrá un total de $2^{13}/2^3 = 2^{10} = 1024$ conjuntos, por lo que la interpretación será la siguiente:

etiqueta	conjunto	byte
17 bits	10 bits	5 bits

c) En el caso de la memoria caché de instrucciones no es posible realizar este solapamiento, ya que se dispone de 16 bits que no requieren traducción en la dirección virtual, pero hacen falta 17 bits para identificar el bloque más el desplazamiento de caché. En el caso de la caché de datos sí es posible solapar la traducción en TLB con el acceso a caché, ya que solo hacen falta 15 bits para identificar el conjunto y el desplazamiento en caché y, como ya se ha mencionado, se dispone de 16 bits de la dirección física en cuanto se dispone de la dirección virtual.

d) Para calcular las tasas de aciertos de las TLB es necesario estudiar la traza de ejecución del programa y distinguir los accesos a diferentes páginas de memoria:

La traza de ejecución viene determinada por la dirección en que se encuentra cada instrucción del código y cada dato al que se accede, es decir, se trata de la siguiente:

```
DIR:  0, 4, 8, 12, 16, 20,
      24, 0x10000, 28, 0x14000, 32, 36, 0x20000, 40, 44, 0x24000, 48, 52, 56,
      24, 0x10004, 28, 0x14004, 32, 36, 0x20004, 40, 44, 0x24004, 48, 52, 56,
      24, 0x10008, 28, 0x14008, 32, 36, 0x20008, 40, 44, 0x24008, 48, 52, 56,
      ..  ......  ..  ......  ..  ..  .......  ..  ..  .......  ..  ..  ..
      24, 0x100FC, 28, 0x140FC, 32, 36, 0x200FC, 40, 44, 0x240FC, 48, 52, 56,
      60, ...
```

Observando únicamente las direcciones que corresponden a instrucciones (desde la 0 hasta la 20 en el preámbulo, desde la 24 a la 56 en el bucle y las siguientes a la 56 al final) se determina que todo el código se aloja en la misma página (la página virtual 0), por lo que únicamente se producirá un fallo en la TLB de instrucciones, mientras que el número de accesos a instrucciones será de $6 + 9 \times 64 = 582$ accesos.

Observando ahora los accesos a datos, estos se producen en las direcciones `0x10000`, `0x14000`, `0x20000`, `0x24000` y repeticiones sucesivas de estas cuatro direcciones incrementadas en 4 cada nueva iteración. Por otro lado, puesto que las páginas utilizadas son de 2^{16} bytes, la página virtual 1 contiene las direcciones `0x00000010000` a `0x0000001FFFF`, es decir, todas las utilizadas en los accesos del programa a las direcciones `0x10000` y sucesivas y `0x14000` y sucesivas. La página virtual 2 contiene las direcciones `0x00000020000` a `0x0000002FFFF`, es decir, todas las utilizadas en los accesos del programa a las direcciones `0x20000` y sucesivas y `0x24000` y sucesivas. Por lo tanto, únicamente se accede a dos páginas de datos diferentes, por lo que se producirán 2 fallos en la TLB de datos. El número de accesos a datos es de $4 \times 64 = 256$ accesos.

En resumen, las tasas de acierto de ambas TLB es la siguiente:

$Hr_{TLB\_I} = (582 - 1)/582 = 0,998 \rightarrow 99,8\,\%$

$Hr_{TLB\_D} = (256 - 2)/256 = 0,992 \rightarrow 99,2\,\%$

e) Utilizando las mismas trazas de ejecución ya descritas en el apartado anterior y considerando que en cada bloque de caché caben 8 palabras (ocho instrucciones u ocho datos), se observa que todas las instrucciones utilizadas (15 consecutivas) se ubican en dos bloques de caché mientras que los datos utilizados (256 direcciones diferentes) requieren $256/8 = 32$ bloques de caché.

El acceso a instrucciones provocará 2 fallos de caché (forzosos), mientras que el acceso a datos requiere un análisis algo más detallado. En primer lugar, puesto que la caché de datos es asociativa por conjuntos de ocho, no es necesario comprobar posibles coincidencias de datos en el mismo bloque de caché (fallos por conflicto). Por otro lado, es necesario distinguir entre lecturas y escrituras, dado que se emplea una política de escritura inmediata que no realiza actualización de la caché en los fallos de escritura. De los 256 accesos a datos la mitad son lecturas, que solo provocarán fallos forzosos ($128/8 = 16$ fallos) y la otra mitad son escrituras, que siempre se completarán con fallo de caché (128 fallos).

Ya se dispone de toda la información para calcular las tasas solicitadas:

$Hr_{McaI} = (582 - 2)/582 = 0,996 \rightarrow 99,6\,\%$

$Hr_{McaD} = (256 - 16 - 128)/256 = 0,4375 \rightarrow 43,75\,\%$

f) Se calcula partir de los tiempos indicados en el enunciado y teniendo en cuenta únicamente los accesos a datos.

El tiempo de traducción es el correspondiente a dos fallos de traducción y 254 aciertos. Por otro lado, ya que se solapa la traducción en TLB de datos con el acceso a la memoria caché de datos, se considerará únicamente el tiempo de traducción debido a los fallos, es decir:

$t_{trad} = 2 \times (2\ ns + 2 \times 60\ ns) = 244\ ns$

El tiempo de acceso a la información se obtendrá sabiendo que en caso de fallo de lectura se contabiliza únicamente el acceso a la primera palabra (*out of order fetch*) y que en las escrituras, por utilizar política inmediata, se invierte el tiempo de acceso a la memoria principal:

$$\begin{aligned} t_{accInfo} &= t_{lecturas} + t_{escrituras} = 112 \times t_{aciertoLectura} + 16 \times t_{falloLectura} + 128 \times t_{escritura} \\ &= 112 \times 3\ ns + 16 \times (3 + 60)\ ns + 128 \times 60\ ns = 9.024\ ns \end{aligned}$$

Por lo que:

$t_{accTotal} = t_{trad} + t_{accInfo} = 244\ ns + 9.024\ ns = 9.268\ ns$

$\overline{t}_{acc} = 9.268\ ns/256\ accesos = 36,2\ ns$

1 Sea un computador cuyo procesador dispone de un pipeline de 7 etapas en el que se quiere ejecutar un fragmento de programa que calcula la suma de cada fila de una matriz, dejando el resultado en otro vector.

El pipeline del procesador introduce ciclos de parada para resolver las dependencias de control, utiliza bifurcación retardada (*delay-slot-2*), dispone de todo tipo de mecanismos de adelantamiento y emplea las siguientes etapas:

- E1: Traducción en TLBi e incremento del PC
- E2: Acceso a McaI para fetch
- E3: Decodificación, lectura de registros, evaluación de la condición y cálculo de la dirección destino en las instrucciones de salto
- E4: Ejecución y cálculo de la dirección para las instrucciones load y store
- E5: Traducción en TLBd
- E6: Acceso a McaD para lectura o escritura de datos
- E7: Escritura en registros

A continuación se muestra el código que se desea ejecutar:

```
                                ;    for (i=0; i++; i<M)
                                ;        for (j=0; j++; j<N)
                                ;            v[i] += mat[i][j];
(1) loop1: add r5, r0, r0       ; r5 acumulador
(2)        add r21,r11, r0      ; r11 y r21 cont. filas
(3) loop2: ld  r1, r10, r0      ; r10 dir matriz, r1 elemento
(4)        add r5, r5, r1
(5)        add r10, r10, 4
(6)        sub r21, r21, 1
(7)        bnz r21, $loop2
(8)        st  r5, r20, 0       ; r20 dir vect
(9)        add r20, r20, 4
(10)       sub r12, r12, 1      ; r12 cont. columnas
(11)       bnz r12, $loop1
```

Se pide lo siguiente:

a) Determine si el programa puede ejecutar correctamente y, si no fuese así, describa qué modificaciones debería realizar para lograr el funcionamiento previsto.

b) Determine en qué instrucciones se producen ciclos de parada por dependencias de datos o por dependencias de control, especificando en cada caso cuántos ciclos de parada se emplean.

c) Calcule el nº de ciclos invertido en la ejecución del programa, así como su CPI. Considere que la matriz consta de 100 filas y 100 columnas.

d) Reordene el código para optimizar el CPI y calcule de nuevo el CPI que se obtiene.

SOLUCIÓN

a) No puede ejecutar correctamente ya que tras cada salto se ejecutarían dos instrucciones más por la bifurcación retardada. Deben introducirse 4 instrucciones `nop`, dos tras cada salto condicional.

b) Hay dependencias de datos entre las instrucciones 3 y 4 por el registro `r1`, así como entre las instrucciones 6 y 7 por el registro `r21` y entre las instrucciones 10 y 11 por el registro `r12`. Además hay dos dependencias de control, las instrucciones 7 y 11, por los saltos condicionales.

La primera dependencia, entre las instrucciones 3 y 4 por el registro `r1`, provoca dos parones, ya que el dato se obtiene en la etapa 6, en el acceso a la cache datos, y se necesita en la etapa 4, a la entrada de la ALU.

Cada una de las otras dos dependencias de datos, entre las instrucciones 6 y 7 por el registro `r21` y entre las instrucciones 10 y 11 por el registro `r12`, provoca un parón, ya que el dato se obtiene en la etapa 4, tras la ejecución en la ALU, y se necesita en la etapa 3, para calcular la condición del salto.

Cada dependencia de control genera dos parones si no tiene instrucciones útiles en el *delay-slot*. Esto se debe a que tanto la dirección como la condición se calculan en la etapa 3 y se necesitarían en la 1, para saber cual es la próxima instrucción.

```
                                ; Ciclos
 (1) loop1: add r5, r0, r0      ;   1
 (2)        add r21,r11, r0     ;   1
 (3) loop2: ld  r1, r10, r0     ;   1 + 2h
 (4)        add r5, r5, r1      ;   1
 (5)        add r10, r10, 4     ;   1
 (6)        sub r21, r21, 1     ;   1 + 1h
 (7)        bnz r21, $loop2     ;   1 + 2h
 (7')       nop                 ;
 (7'')      nop                 ;
 (8)        st  r5, r20, 0      ;   1
 (9)        add r20, r20, 4     ;   1
(10)        sub r12, r12, 1     ;   1 + 1h
(11)        bnz r12, $loop1     ;   1 + 2h
(11')       nop                 ;
(11'')      nop                 ;
```

c) El CPI se calcula como el cociente entre los ciclos necesitados para ejecutar el programa y las instrucciones útiles ejecutadas, es decir:

$$\frac{100\times(2+100\times(5+5h)+4+3h)}{100\times(2+100\times5)+4)} = 1,99$$

d) Una posible reordenación, que suprime la mayor parte de los parones, sería la siguiente:

```
                                ; Ciclos
 (1) loop1: add r5, r0, r0      ;   1
 (2)        add r21,r11, r0     ;   1
 (3) loop2: ld  r1, r10, r0     ;   1
 (6)        sub r21, r21, 1     ;   1
 (5)        add r10, r10, 4     ;   1
 (7)        bnz r21, $loop2     ;   1 + 1h
 (4)        add r5, r5, r1      ;   1
 (7')       nop                 ;
(10)        sub r12, r12, 1     ;   1
 (8)        st  r5, r20, 0      ;   1
(11)        bnz r12, $loop1     ;   1 + 1h
 (9)        add r20, r20, 4     ;   1
(11')       nop                 ;
```

Esta reordenación suprime los parones debido a las dependencias de datos, ya que pone intrucciones suficientes como para que se puedan adelantar los resultados, y ubica una instrucción útil tras cada salto, por lo que sólo queda un `nop`, es decir, un parón tras cada salto condicional. El CPI queda como:

$$\frac{100\times(2+100\times(5+1h)+4+1h)}{100\times(2+100\times5)+4)} = 1,2$$

Siendo más agrasevimos en la reordenación podemos dejar:

```
                                ; Ciclos
 (1) loop1: add r5, r0, r0      ;   1
 (2)        add r21,r11, r0     ;   1
 (6) loop2: sub r21, r21, 1     ;   1
 (3)        ld  r1, r10, r0     ;   1
 (7)        bnz r21, $loop2     ;   1
 (5)        add r10, r10, 4     ;   1
 (4)        add r5, r5, r1      ;   1
(10)        sub r12, r12, 1     ;   1
 (8)        st  r5, r20, 0      ;   1
```

```
(11)        bnz r12, $loop1     ;   1 + 1h
 (9)        add r20, r20, 4     ;   1
(11')       nop                 ;
```

El CPI queda como:

$\frac{100\times(2+100\times(5)+4+1h)}{100\times(2+100\times5)+4)} = 1,0$

2 Sea un multiprocesador de memoria compartida con 16 procesadores, conectados a través de un bus de ciclo partido a la memoria principal. Cada procesador dispone de una memoria cache privada de datos, que por simplificar se va a considerar que tiene sólo cuatro bloques, es de ubicación directa y que cada bloque tiene sólo dos palabras.

Para mantener la coherencia usa la política de invalidación MESI, con cuatro estados, sobre una implementación *snoopy*

Indique las acciones que desencadenan las siguientes operaciones y qué cambios se producen. Suponga que cada operación parte del estado inicial mostrado en la siguiente figura.

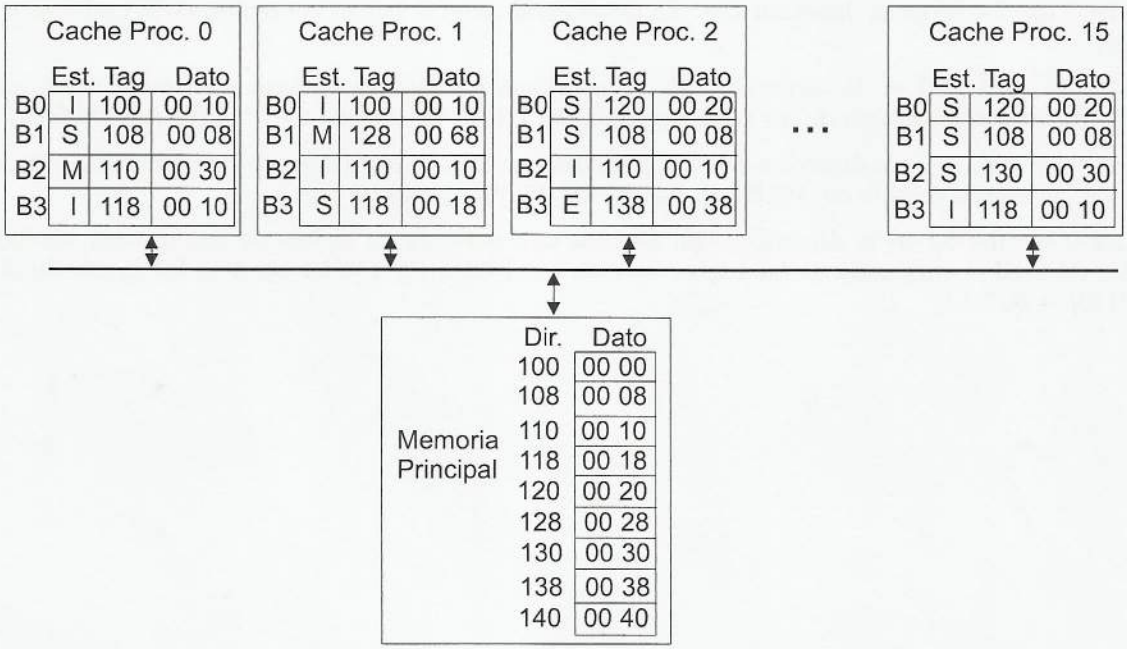

Figura 1 Multiprocesador con coherencia de caches basada en MESI

a) Procesador 0 lee el dato de la dirección 120.

b) Procesador 2 lee el dato de la dirección 118.

c) Procesador 0 escribe 80 en la dirección 110.

d) Procesador 0 escribe 80 en la dirección 120.

e) Procesador 15 escribe 80 en la dirección 130.

f) Procesador 2 lee el dato de la dirección 110.

g) Procesador 2 escribe 80 en la dirección 138.

h) Procesador 1 escribe 80 en la dirección 130.

i) Procesador 0 lee el dato de la dirección 138.

j) Procesador 0 escribe 80 en la dirección 130.

SOLUCIÓN

a) Procesador 0 lee el dato de la dirección 120. Genera una petición de lectura en el bus, dirección 120. El contenido de la cache será P0.B0 (S 120 00 20)

b) Procesador 2 lee el dato de la dirección 118. Genera una petición de lectura en el bus, dirección 118. El contenido de la cache será P2.B3 (S 118 00 18)

c) Procesador 0 escribe 80 en la dirección 110. No genera tráfico en el bus. El contenido de la cache será P0.B2 (M 110 00 80)

d) Procesador 0 escribe 80 en la dirección 120. Genera una petición en el bus de lectura con invalidación, dirección 120. Al final el contenido de las caches cambia en: P0.B0 (M 120 00 80) P2.B0 (I 120 00 20) P15.B0 (I 120 00 20)

e) Procesador 15 escribe 80 en la dirección 130. Genera una petición de invalidación sobre la dirección 130. El contenido de la cache será P15.B2 (M 130 00 80)

f) Procesador 2 lee el dato de la dirección 110. Genera una petición de lectura en el bus, dirección 110. Al final el contenido de las caches cambia en: P0.B2 (S 110 00 30) P2.B2 (S 110 00 30) y la memoria M[110] = 00 30

g) Procesador 2 escribe 80 en la dirección 138. No genera tráfico en el bus.El contenido de la cache será P2.B3 (M 138 00 80)

h) Procesador 1 escribe 80 en la dirección 130. Genera una petición de lectura con invalidación sobre la dirección 130. Al final el contenido de las caches cambia en: P1.B2 (M 130 00 80) P15.B2 (I 130 00 30)

i) Procesador 0 lee el dato de la dirección 138. Genera una petición de lectura en el bus, dirección 138. Al final el contenido de las caches cambia en: P0.B3 (S 138 00 38) P2.B3 (S 138 00 38)

j) Procesador 0 escribe 80 en la dirección 130. Genera una petición en el bus de lectura con invalidación, dirección 130. Al final el contenido de las caches cambia en: P0.B2 (M 130 00 80) P15.B2 (I 130 00 30) y la memoria M[110] = 00 30

1 *Sea una CPU de 32 bits de ancho de palabra y 2000 MIPS de capacidad de ejecución. Se desea conectar a este procesador una unidad de red que tiene una velocidad de transferencia de* $640 \cdot 10^6$ *bits/s y un registro de datos de una palabra. Se supone un funcionamiento mediante interrupciones donde la Secuencia de Reconocimiento de Interrupción tiene una duración de 5 ns y su rutina de interrupción 20 ns. Calcule:*

a) El tiempo que transcurre entre dos solicitudes de interrupción consecutivas (o período de las solicitudes de interrupción) .

b) A partir del resultado del apartado anterior, determine cuántos periféricos podrían operar simultáneamente con esa configuración.

Posteriormente, se añade al módulo de E/S un buffer de 400 bytes. La nueva rutina de interrupción para operar ahora con este buffer tiene una duración de 220 ns. Calcule para esta nueva situación:

c) La frecuencia de las solicitudes de interrupción, Int/s

d) El nuevo número máximo de unidades de red que podrían operar simultáneamente.

e) Indique brevemente cómo influye el uso del buffer en los resultados obtenidos.

SOLUCIÓN

a) Se produce una interrupción por cada buffer, que es en este caso el registro de datos único de capacidad una palabra, 4 bytes, a la velocidad de transferencia del periférico, consecuentemente:

$$1\ Int/buffer \times 1buffer/4B \times 1B/8bits \times 640 \cdot 10^6\ bits/s = 20 \cdot 10^6 Int/s$$

o lo que es lo mismo una interrupción cada 50 ns, $T_{Int} = 50ns$. En este tiempo se pueden ejecutar 100 instrucciones.

b) El servicio de cada interrupción consume el tiempo fijo asociado a la Secuencia de Reconocimiento de Interrupción (SRI), $t_{SRI} = 5ns$, más el tiempo de la rutina de interrupción propiamente dicha, $t_{Rut\_Int} = 20ns$. Cada interrupción consume:

$$t_{Int} = t_{SRI} + t_{Rut\_Int} = 5ns + 20ns = 25ns$$

De este valor y del periodo entre interrupciones se obtiene que sólo cabría el funcionamiento simultáneo de dos unidades del perférico.

c) De nuevo se producirá una interrupción por cada buffer, sólo que ahora tiene una capacidad de 400 bytes, i.e., 100 palabras:

$$1\ Int/buffer \times 1buffer/400B \times 1B/8bits \times 640 \cdot 10^6\ bits/s = 2 \cdot 10^5 Int/s$$

Esta frecuencia de solicitudes de interrupción representa un consumo de acuerdo a la duración de la nueva rutina de servicio, $t_{Rut\_Int'} = 220ns$, y el tiempo total que se consume en cada interrupción, $t_{Int'} = t_{SRI} + t_{Rut\_Int'} = 5ns + 220ns = 225ns$. Por lo tanto el consumo total expresado en su equivalente en instrucciones queda:

$$2 \cdot 10^5 Int/s \times t_{Int'} \times 2instrucciones/1ns = 90MIPS$$

En los 2000 MIPS de capacidad total de ejecución podrían operar 22 periféricos, que representan un consumo total de 1980 MIPS.

d) El uso del buffer de 100 palabras reduce a la centésima parte el número de interrupciones. La nueva rutina de interrupción tiene una duración 205 ns mayor, se supone que debido a que debe ejecutar código extra para operar con el buffer. Sin embargo, el consumo total de CPU disminuye significativamente ya que todo el coste fijo (*overhead*) de cada interrupción se reparte entre las 100 palabras del buffer en lugar de la única palabra de la situación inicial.

2 *Considerando los datos del problema anterior:*

a) Calcule el tiempo de una operación de la red así como el tiempo que se emplea de CPU teniendo en cuenta:

- *Las rutinas de inicio y fin de operación constan de 200 y de 150 instrucciones respectivamente.*
- *Considere un tamaño de bloque de 1.024 bytes y un sólo registro de datos de 32 bits.*

Al computador del problema anterior se le conecta una unidad de disco duro que opera por DMA con las siguientes características:

- *Velocidad de transferencia:* $40 \cdot 10^6$ *bytes/s.*
- *Sectores de 1.024 bytes de capacidad neta.*
- *Tiempo medio de acceso: 4 ms.*
- *Registro de datos de 32 bits.*
- *Las rutinas de inicio y fin de una operación de E/S constan de 100 y de 50 instrucciones respectivamente.*
- *El protocolo de concesión y liberación de los buses dura 2 ns en total.*
- *El ciclo de acceso a memoria dura 2 ns.*

b) Calcule cuántos de estos discos duros podrían operar simultáneamente.

c) Calcule el tiempo total de una operación de lectura de un sector del disco duro así como el tiempo que se emplea de CPU.

Se lee un fichero del disco duro de 10.240 bytes y se transmite simultáneamente por el módulo de red.

d) Calcule la duración total de las operaciones de lectura y transmisión del fichero.

e) Calcule qué porcentaje de la duración total de las operaciones ha podido dedicar la CPU a ejecutar otros procesos.

SOLUCIÓN

a) Se calcula el tiempo total y el de CPU de una operación de la red teniendo en cuenta que a 2.000 MIPS una instrucción se procesa en 0,5 ns.

$$\begin{aligned} t_{op\_red} &= t_{ini} + t_{trans} + t_{última-interrupción} = t_{ini} + t_{trans} + (t_{sri} + t_{int} + t_{fin}) = \\ &= 100\ ns + \frac{1.024\ bytes \cdot 8\ bits/byte}{640 \cdot 10^6\ bits/s} + (5\ ns + 20\ ns + 75\ ns) = 13.000\ ns \\ t_{cpu\_red} &= t_{ini} + N_{interrupciones} \times t_{int} + t_{fin} = \\ &= 100\ ns + \frac{1.024\ bytes}{4\ bytes} \times (5\ ns + 20\ ns) + 75\ ns = 6.575\ ns \end{aligned}$$

b) Se calcula la capacidad de procesamiento que se le quita al procesador al conceder robos de ciclos.

Hay que tener en cuenta que un robo de ciclo detiene al procesador durante la duración del protocolo de concesión y liberación de los buses y el ciclo de acceso a memoria. Además se calculan cuantas instrucciones deja de ejecutar el procesador por cada robo.

$$\begin{aligned} t_{robo} &= 2\ ns + 2\ ns = 4\ ns \\ I_{robos} &= 4\ ns \times 2\ instr/ns = 8\ instr \end{aligned}$$

$$\begin{aligned} CP_{HD} &= Freq_{robos} \times I_{robos} = \frac{40 \cdot 10^6\ Byte/s \cdot 8\ bits/Byte}{32\ bits} \times 8\ instr = \\ &= 10 \cdot 10^6\ Hz \times 8\ instr = 80 \cdot 10^6\ instr/s = 80\ MIPS \end{aligned}$$

Luego podrían operar hasta 25 discos (2.000/80) simultáneamente.

c) Se calcula el tiempo total y el de CPU de una operación del disco.

$$\begin{aligned} t_{op\_HD} &= t_{ini} + t_{acc} + t_{trans} + t_{último-robo} + t_{interrupción} = t_{ini} + t_{acc} + t_{trans} + t_{último-robo} + (t_{sri} + t_{int}) = \\ &= 50\ ns + 4 \cdot 10^6\ ns + \frac{1.024\ bytes}{40 \cdot 10^6\ bytes/s} + 4\ ns + (5\ ns + 25\ ns) = 4.025.684\ ns \\ t_{cpu\_HD} &= t_{ini} + N_{robos} \times t_{robos} + t_{interrupción} = \\ &= 50\ ns + \frac{1.024\ bytes}{4\ bytes} \times 4\ ns + 30\ ns = 1.104\ ns \end{aligned}$$

d) El fichero ocupa 10 sectores en el disco y se transmite mediante 10 bloques de red. Será necesario leer el primer sector del disco y a partir de ese momento se lee el segundo a la vez que se transmite el primero. Así el tiempo total será el de 10 operaciones del disco y la última de la red.

$$t_{op\_total} = 10 \cdot t_{op\_HD} + t_{op\_red} = 10 \cdot 4.025.684\ ns + 13.000\ ns = 40.269.840\ ns$$

e) El tiempo que ha estado ocupada la CPU es el correspondiente a 10 operaciones del disco y 10 operaciones de la red.

$$t_{CPU\_total} = 10 \cdot t_{CPU\_HD} + 10 \cdot t_{CPU\_red} = 10 \cdot 1.104\ ns + 10 \cdot 6.575\ ns = 76.790\ ns$$

Así el porcentaje de CPU ocupada es:

$$\%CPU_{ocupada} = \frac{t_{CPU\_total}}{t_{op\_total}} \times 100 = \frac{76.790}{40.269.840} \times 100 = 0,19\,\%$$

El el porcentaje de CPU que se ha podido dedicar a ejecutar otros procesos es: 99,81 %.

1 *Sea un procesador con tamaño de palabra de 64 bits, direcciones físicas de 40 bits y direccionamiento a nivel de byte, cuyo sistema de memoria está compuesto por una memoria principal y un único nivel de memoria caché con dos cachés separadas, para instrucciones (McaI) y para datos (McaD), cuyas características son las siguientes:*

- *Capacidad 64KB, bloques de 64B y tiempo de acceso de 2 ns*
- *Ubicación asociativa por conjuntos de 4 bloques, política de lectura out of order fetch y política de reemplazo LRU*
- *Para la caché de datos, la política de escritura utilizada es aplazada con actualización (CBWA). En los fallos de escritura, primero se lleva el bloque a la caché y después se realiza la escritura en caché*
- *El tiempo empleado en transferir un bloque entre Mp y Mca es 65 ns, y el tiempo empleado para leer o escribir una palabra en Mp es 50 ns.*

En este procesador se ejecuta un programa, del que se ha extraído el siguiente fragmento. Las variables i y j están asignadas a registros y las matrices, todas ellas de 64 x 32 elementos de 64 bits (una palabra), están almacenadas en Mp por filas. Observe que se accede a todos los elementos de A y C, pero solo a una parte de B.

```
for (i=0; i<64; i++)
    for (j=0; j<32; j++)
        C[i][j] = A[i][j]*B[j][0];
```

Sabiendo que la matriz A está almacenada en Mp a partir del bloque 512, y a continuación se encuentran C y B, en este orden.

a) *Calcule el número de bloques que ocupan los elementos de las matrices a los que hace referencia el código anterior. Indique si la caché de datos tiene capacidad suficiente para albergarlos.*

b) *Especifique cómo interpreta la McaD las direcciones físicas: campos en que se descompone, el significado de cada uno y su longitud, y en qué conjuntos se alojarán los elementos de la matrices a los que hace referencia el código.*

c) *Calcule el número de aciertos y fallos que se producen en la McaD en el acceso a las tres matrices, así como la tasa de aciertos.*

d) *Indique razonadamente qué referencias del código anterior muestran proximidad espacial y cuáles proximidad temporal.*

SOLUCIÓN

a) Cada matriz ocupa $(64\ filas \times 32\ columnas \times 8B/elemento) = 2^{14}B$ y cada bloque tiene capacidad para 64B, por lo que cada matriz ocupa $2^{14}B/2^6B/bloque = 2^8 = 256\ bloques$ de Mp.

En el caso de la matriz B únicamente se hace referencia a los primeros 32 elementos de su primera columna, por lo que de los 256 bloques que ocupa en Mp únicamente se hace referencia a 32 bloques.

En total, el número de bloques pedido es: $2 \times 256 + 32 = 544\ bloques$

La McaD tiene capacidad para: $64KB/64B/bloque = 2^{10} = 1.024\ bloques$, y por lo tanto tiene capacidad suficiente para albergar los 544 bloques.

b) Al ser la política de ubicación de la McaD asociativa por conjuntos de 4 bloques (cada uno de 64B), está compuesta por: $2^{10}\ bloques/2^2\ bloques/conjunto = 2^8 = 256\ conjuntos$

Por lo tanto, la McaD interpreta las direcciones físicas de 40 bits de la siguiente forma:

etiqueta	conjunto	byte en bloque
26 bits	8 bits	6 bits

El campo *conjunto* indica el conjunto de la caché donde le corresponde estar ubicado al bloque de Mp al que se hace referencia, el campo *byte* indica el byte concreto al que se hace referencia, dentro del bloque. Finalmente, el campo *etiqueta* indica cuál de los posibles bloques de Mp a los que corresponde ir al conjunto seleccionado es al que se hace referencia.

El primer bloque de Mp ocupado por la matriz A es el bloque 512, que se ubicará en el conjunto 512 *mod* 256 = 0, el segundo en el conjunto 1, y así sucesivamente hasta el último, que se ubicará en el conjunto 255, ocupando así uno de los bloques de cada uno de los conjuntos de la caché. Como la matriz C está almacenada

en Mp a continuación de A se ubicará en los mismos conjuntos que ésta, ocupando otro de los bloques de cada conjunto.

Por último, para calcular en qué conjuntos se ubicarán los 32 bloques de C, a los que se hace referencia, es necesario calcular previamente cuántos bloques ocupa cada fila:

$(32\ elementos/fila \times 8B/elemento)/64B/bloque = 4\ bloques/fila$

En consecuencia, dichos bloques se ubicarán en los conjuntos 0, 4, 8, 12, ..., 124 de caché, ocupando un tercer bloque de dichos conjuntos.

c) Los únicos fallos que se producirán son de primera referencia, ya que las matrices A y C, junto con los 32 bloques de C caben en la caché, y no se producen conflictos por un mismo conjunto, puesto que cada uno está compuesto por 4 bloques. Como se hace referencia a un total de 544 bloques diferentes, se producirán 544 fallos de primera referencia, en el primer acceso a la primera palabra de cada bloque.

Para calcular el número de aciertos, basta con restar los fallos del número total de accesos:

$3 \times 64 \times 32 = 6.144\ accesos - 544\ fallos = 5.600\ aciertos,$

por lo que la tasa de aciertos es: $(5.600\ aciertos/6.144\ accesos) \times 100 = 91,14\,\%$

d) Las referencias a las matrices A y C presentan proximidad espacial, concretamente secuencial, ya que corresponden a direcciones consecutivas de memoria. Sin embargo, las referencias a B no presentan dicho tipo de proximidad, sino temporal ya que para cada nuevo valor de la variable "i"se vuelve a acceder a los mismos elementos de B.

2 *Considere ahora que este procesador corresponde a un computador con memoria virtual paginada y las siguientes características:*

- *Páginas de 4KB y tres niveles de tablas de páginas, en las que cada entrada ocupa una palabra*
- *TLBs separadas para instrucciones y datos cuyo tiempo de acceso es 1 ns*

a) *Indique razonadamente cuántas entradas debería tener como mínimo la TLB de datos para que se produzcan únicamente fallos de primera referencia en la traducción de los datos del código del enunciado.*

b) *Calcule los tiempos mínimo y máximo de acceso al sistema de memoria de este computador, indicando claramente en qué condiciones se produce cada uno, suponiendo que no se produce fallo de página. Considere tanto accesos de lectura como de escritura.*

SOLUCIÓN

a) Se calcula, en primer lugar, el número de páginas que ocupan las matrices. La matriz A está en el bloque de Mp 512 (2^9), esto es en la dirección de Mp $2^9 \times 2^6 = 2^{15}$, dirección alineada a página ya que los 12 bits menos significativos son cero (las páginas son de $2^{12}B$).

$(64 \times 32 \times 8B)/4KB/pagina = 2^{14}/2^{12} = 4$ páginas cada matriz.

Los elementos de las matrices A y C se recorren una sola vez, en el mismo orden en el que están almacenadas en memoria, por lo que bastaría con tener una entrada en la TLB por cada una de ellas, la correspondiente a la página utilizada en cada momento, puesto que al cambiar de página se podría reemplazar la entrada correspondiente a la página anterior, que ya no se vuelve a utilizar.

En cuanto a la matriz B se necesitan dos entradas, las correspondientes a las dos primeras páginas, en las que se encuentran los primeros 32 elementos de su primera columna, a los que se hace referencia en cada iteración del bucle controlado por la variable *i*.

En consecuencia, para obtener únicamente fallos de primera referencia en la TLB de datos bastaría con que ésta tuviese cuatro entradas.

b) El tiempo mínimo de acceso se producirá en caso de acierto en TLB y en caché. Como no se puede solapar el acceso a ambos dispositivos, ya que las páginas son de 4KB ($2^{12}B$) y con los 12 bits que no se traducen no se pueden direccionar los 2^8 conjuntos, más el byte dentro del bloque ($2^6B/bloque$):

$T_{min} = T_{TLB} + T_{Mca} = 1 + 2 = 3\ ns$

El tiempo máximo corresponde a un acceso con fallo de traducción en la TLB, con la correspondiente consulta a los tres niveles de tablas residentes en Mp. Una vez realizada la traducción, el tiempo máximo de

acceso a la información corresponde a un acceso con fallo en caché y el bloque a reemplazar ha sido modificado.

Si el acceso es de escritura, el tiempo máximo es el siguiente:

$T_{max}(E) = T_{TLB} + T_{TP's} + T_{Mca} + T_{blqMca \rightarrow Mp} + T_{blqMp \rightarrow Mca} + T_{Mca} = 1 + 3 \times 50 + 2 + 65 + 65 + 2 = 285\ ns$

Obsérvese que, según el enunciado, cuando se produce un fallo de escritura primero se envía a Mca el bloque que produce el fallo y a continuación se realiza el acceso de escritura en la Mca.

En caso de lectura, el tiempo máximo de traducción es el mismo, cambiando únicamente el tiempo máximo de acceso a la información. Ya que la caché tiene política de lectura *out of order fetch*, el tiempo pedido es:

$T_{max}(L) = T_{TLB} + T_{TP's} + T_{Mca} + T_{blqMca \rightarrow Mp} + T_{Mp} = 1 + 3 \times 50 + 2 + 65 + 50 = 268\ ns$

3 *En la ejecución del programa completo, que incluye el código anterior, se han obtenido los siguientes valores en las referencias a datos:*

- *El 70 % son de lectura*
- *Tasa de aciertos de la McaD: 93 % y tasa de aciertos de la TLBD: 95 %*
- *La probabilidad de reemplazar un bloque modificado es del 20 %*

a) Calcule el tiempo medio de acceso a los datos, suponiendo que no se producen fallos de página.

b) Indique cómo cambiaría el tiempo calculado en el apartado anterior si la política de escritura utilizada fuese inmediata sin actualización (WTWNA) y calcule la ganancia que se obtendría utilizando esta política.

SOLUCIÓN

a) El tiempo medio de acceso a datos se puede calcular, en este caso, como la suma de los tiempos medios de traducción y acceso a la información: $\overline{T}_{acc} = \overline{T}_{trad} + \overline{T}_{inf}$

$\overline{T}_{trad} = T_{TLB} + (1 - Hr_{TLB}) \times T_{TP's} = 1 + 0,05 \times (3 \times 50) = 8,5\ ns$

Para calcular el tiempo medio de acceso a la información hay que tener en cuenta si el acceso es de lectura o de escritura y si el el bloque a reemplazar, en caso de fallo en caché, está modificado:

$\overline{T}_{inf} = Pr_L \times \overline{T}_{inf}L + Pr_E \times \overline{T}_{inf}E$

$\overline{T}_{inf}L = T_{Mca} + (1 - Hr_{Mca}) \times [Pr_{Mod} \times T_{blqMca \rightarrow Mp} + T_{Mp}] = 2 + 0,07 \times [0,2 \times 65 + 50] = 6,41\ ns$

$\overline{T}_{inf}E = T_{Mca} + (1 - Hr_{Mca}) \times [Pr_{Mod} \times T_{blqMca \rightarrow Mp} + T_{blqMp \rightarrow Mca} + T_{Mca}] =$

$= 2 + 0,07 \times [0,2 \times 65 + 65 + 2] = 7,6\ ns$

$\overline{T}_{inf} = 0,7 \times 6,41 + 0,3 \times 7,6 = 6,77\ ns$

$\overline{T}_{acc} = 8,5 + 6,77 = 15,27\ ns$

b) Si la política de escritura fuese WTWNA la probabilidad de reemplazar un bloque modificado sería cero, ya que los accesos de escritura se realizan a la vez en la caché y en la Mp, por lo que la información en ambos niveles es siempre coherente. El tiempo de acceso en escritura correspondería, en este caso, al tiempo de Mp, ya que ésta es más lenta que la caché.

Ya que el tiempo medio de traducción no cambiaría con respecto al apartado anterior, únicamente hay que recalcular los tiempos medios de acceso a la información:

$\overline{T}_{inf}L = T_{Mca} + (1 - Hr_{Mca}) \times T_{Mp} = 2 + 0,07 \times 50 = 5,5\ ns$

$\overline{T}_{inf}E = max(T_{Mca}, T_{Mp}) = 50\ ns$

Teniendo en cuenta el porcentaje de lecturas y escrituras:

$\overline{T}_{inf} = 0,7 \times 5,5 + 0,3 \times 50 = 18,85\ ns$

Y sumando el tiempo medio de traducción:

$\overline{T}_{acc} = 8,5 + 18,85 = 27,35\ ns$

Como se puede comprobar, la utilización de esta política de escritura no proporciona ganancia alguna sino todo lo contrario, ya que el tiempo medio de acceso es mayor que utilizando CBWA.

1 [2 puntos] Responda *razonadamente* a las las siguientes cuestiones sobre el sistema de E/S:

a) ¿Por qué surge la necesidad de las diferentes técnicas de E/S? Señale cuál o cuáles de ellas implican una modificación en el procesador y en qué consiste.

b) Explique en qué consiste el sobrecoste u *overhead* en tiempo asociado a las interrupciones y de qué manera el tamaño del *buffer* que se emplea en el módulo de E/S puede ayudar a reducirlo.

c) ¿De qué manera cree que el tamaño del *buffer* del módulo de E/S afecta al tiempo de CPU que se consume para la operación de E/S en el funcionamiento mediante DMA?

d) En la asignación de las prioridades de interrupción se suele otorgar más prioridad al periférico que interrumpe con más frecuencia, lo que dependerá de su velocidad de transferencia y del tamaño de su *buffer*. Indique algunas posibles excepciones a este criterio.

SOLUCIÓN

a) Las técnicas de E/S vienen motivadas, fundamentalmente, por el hecho de que los periféricos son mucho más lentos que el resto del sistema al que se conectan y porque, además, tienen un funcionamiento simultáneo e independiente. Estas características dan lugar a la sabida necesidad de sincronización entre datos –al final, entre *búfferes*–. En E/S programada la CPU no se puede utilizar para otra labor distinta durante toda la operación de E/S. Tanto en interrupciones como en DMA se trata de liberar al procesador de la fase de sincronización. Ambas significan modificaciones en la arquitectura –mecanismo de interrupciones y soporte del arbitraje del bus del sistema en el DMA– con respecto a la usada para la E/S programada.

b) Las interrupciones tienen un coste fijo (en tiempo de CPU) que se debe fundamentalmente a la secuencia de reconocimiento de interrupción y al código que se ejecuta para salvaguardar y reponer el estado de la ejecución y al que corresponde al paso y actualización de los parámetros. Este coste fijo –con pequeñas variaciones dependiendo del tamaño del buffer– se repartirá entre el número de interrupciones que se produzca (i.e., en el caso de un buffer de capacidad "infinita" el sobrecoste medio sería cero).

c) En el caso del DMA, el coste fijo cada vez que se ceden y devuelven los buses es el de tiempo que corresponde al procolo correspondiente y que es del orden de unos pocos nanosegundos. El tamaño del buffer determinaría el número de DMAs, y el pequeño sobrecoste citado se repartiría entre el número de palabras que contuviese, lo que representaría un ahorro del tiempo en que la CPU cede los buses, pero de manera mucho menos significativa que en el uso anterior en el caso de interrupciones.

d) Otro criterio pudiera ser la "importancia" del periférico, p.ej., un detector de caída de tensión o una alarma de pánico en una central nuclear –que ojalá jamás se diese–. También a veces se asigna mayor prioridad a periféricos con menor frecuencia de interrupción, pero cuya rutina de servicio es muy breve, p.ej., para aumentar lo que podríamos llamar "reactividad" (o *responsiveness*) frente a las interacciones del usuario.

2 [3 puntos] Sea un procesador capaz de ejecutar 1000 MIPS y un disco duro que transfiere datos a una velocidad de $20 \cdot 10^6$ bytes/s, tiempo de acceso de 5 ms y un registro de datos de 64 bits, 8 bytes, que es el ancho de palabra del computador. Suponga que se opera mediante. interrupciones.

a) Se conectan dos de estos discos duros al procesador. Si la secuencia de reconocimiento de interrupciones, SRI, dura 5 ns y se desea que puedan operar simultáneamente, ¿cuántas instrucciones como máximo se podrán ejecutar en su rutina de interrupción, RTI?

SOLUCIÓN

Tiempo entre interrupciones de cada disco:

$\frac{64bits}{20 \cdot 8 \cdot 10^6 bits/s} = 400ns$, como son dos discos, 200ns.

Tiempo necesario para ejecutar una instrucción:

$\frac{1}{1000MIPS} \Rightarrow 1ns$ por instr.

Atender cada interrupción supone realizar la SRI y ejecutar la RTI, por lo que no podrá ejecutar más de: $\frac{200ns-5ns}{1ns/instr.} = 195$instrucciones

a) Sea uno solo de estos discos duros y una tarjeta de red que funciona mediante interrupciones y que tiene un *buffer* de 256 bits (4 palabras ó 32 bytes), ambos de nuevo conectados al procesador anterior. Si en la RTI del disco se ejecutan 60 instrucciones y en la RTI de la red se ejecutan 50 instrucciones, calcule la velocidad máxima a la que puede operar la red para que puedan trabajar simultáneamente.

SOLUCIÓN

El tiempo empleado en atender las interrupciones del disco y de la red no puede sobrepasar el tiempo disponible. El consumo de cada interrupción viene dado por la duración de la SRI y la RTI, por lo que calculamos el número de interrupciones por segundo que genera el disco y de ahí obtenemos el número máximo de interrupciones que puede generar la red con el buffer empleado, a partir de lo cual hallaremos la velocidad.

$\frac{20 \times 8 \times 10^6 bits/s}{64bits/int} = 2,5 \cdot 10^6 int/s$

$2,5 \cdot 10^6 int/s \cdot (5+60) inst/int + X \cdot 1int/256bits \cdot (5+50) inst/int < 1000 \cdot 10^6 inst/s$

$X = \frac{1000 \cdot 10^6 inst/s - 162,5 \cdot 10^6 inst/s}{\frac{55inst}{256bits}} = 3.898,18 \cdot 10^6 bits/s \Rightarrow \simeq 487MB/s$

3 [5 puntos] Sea un computador de 32 bits con una CPU capaz de ejecutar 1000 MIPS y con un sistema de interrupciones cuya SRI tiene una duración de 5 ns.

a) Este computador tiene conectado un periférico P1 con las siguientes características:

- Velocidad de transferencia: $20 \cdot 10^6$ bytes/s.
- Tiempo medio de acceso: 5 ms.
- Registro de datos de 32 bits.
- Tamaño del bloque: 1.024 bytes.

El módulo de E/S del periférico opera mediante interrupciones con los siguientes parámetros:

- La rutina de programación de una operación de E/S ejecuta 100 instrucciones.
- La rutina de fin de una operación de E/S ejecuta 90 instrucciones.
- En la rutina de servicio de interrupción se ejecutan 50 instrucciones.

a.1) Calcule el número máximo de periféricos como P1 que pueden operar simultáneamente.

a.2) Calcule el tiempo total de una operación de lectura de un bloque.

b) Este computador dispone también de una conexión a un perférico P2 que opera mediante DMA por ráfagas con las siguientes características:

- Velocidad de transmisión de 1 Gigabit por segundo (10^9 bits/s).
- Buffer de 4 registros de datos de 32 bits.
- Las rutinas de inicio y fin de una operación de E/S tienen 100 y 150 instrucciones respectivamente.
- El protocolo de concesión/liberación de buses consume un total de 2 ns.
- El tiempo de un ciclo de acceso a memoria principal es de 10 ns/palabra .
- Bloques de 512 bytes.

b.1) Indique cuántas ráfagas de DMA se efectuarán durante una operación de E/S.

b.2) Indique cuánto durará cada ráfaga de DMA.

b.3) Calcule la duración total de una operación de transferencia de un bloque a través de P2.

c) Calcule la duración total de la transferencia de un archivo de 4 KB desde P1 a P2. Haga las suposiciones que estime oportunas, detallándolas en su contestación.

SOLUCIÓN

a)

a.1) Se calcula la capacidad de proceso que requiere la atención a las interrupciones de un disco y se obtiene el número de discos que pueden operar simultáneamente.

$$\begin{aligned} Capacidad\_de\_proceso &= Frecuencia\_de\_interrupciones \times instrucciones/interrupción = \\ &= \frac{20 \cdot 10^6\ bytes/s}{4\ bytes/interrupción} \times (5+50) instrucciones/interrupción = \\ &= 275 \cdot 10^6 instrucciones/s \end{aligned}$$

$$Número\_de\_discos = \left\lceil \frac{10^9\ instrucciones/s}{275 \cdot 10^6 instrucciones/s/perif} \right\rceil = 3perif$$

a.2) Para realizar la operación de lectura de un bloque, en primer lugar habrá que programar la lectura lo que supone la ejecución de la rutina de inicio de operación.

Tras el tiempo de acceso de 5 ms se comienza la lectura de los 1.024 bytes del sector y finalmente la rutina de tratamiento de interrupción que transfiere el los últimos 4 bytes y la que finaliza la operación.

$$t_{operación} = 100\ ns + 5\ ms + \frac{1.024\ bytes}{20 \cdot 10^6\ bytes/s} + (5+50)\ ns + 90\ ns = 5,051445\ ms$$

b)

b.1) El módulo de E/S solicitará los buses para acceder en DMA una vez por cada buffer. Como es buffer tiene un tamaño de 4 palabras y el bloque de 512 bytes, i.e., 128 palabras, se producirá un total de 32 ráfagas de DMA.

b.2) Como el dispositivo posee un buffer de 4 registros los robos de ciclo se producen en ráfagas de 4 accesos a memoria. Por lo tanto la duración de una ráfaga es $4 \cdot 10\ ns + 2\ ns = 42\ ns$.

b.3) Para transmitir un bloque, en primer lugar se programará la operación, lo que supone ejecutar la rutina de inicio de operación.

Tras el primer DMA se transmiten los 512 bytes del bloque y cuando finaliza la transmisión se ejecutaría la rutina de tratamiento de interrupción correspondiente al fin de operación, aunque no aparece especificada en este caso, lo cual supone un funcionamiento por muestreo o *polling* para que el Sistema Operativo pueda determinar tal circunstancia. La rutina de finalización añadirá 150 instrucciones.

$$t_{operación} = 100\ ns + 42\ ns + \frac{512\ bytes}{125 \cdot 10^6\ bytes/s} + 150\ ns = 4.388\ ns$$

b.4) Del tiempo anterior de operación, la CPU estará ocupada –según el criterio seguido en las clases del tema, puesto que en puridad es sólo el tiempo en se cede el bus del sistema– el correspondiente a la suma de los DMAs, más el correspondiente a las rutinas de inicio y fin.

$$\begin{aligned} t_{ocupCPU} &= 100\ ns/rutina\_inicio + 32\ DMA \times 42\ ns/DMA \\ &+ 150\ ns/rutina\_fin = 1.594\ ns \end{aligned}$$

Y el tiempo de CPU para otros procesos será: 4.388 ns - 1.594 ns = 2.794 ns

c) El tamaño del archivo, 4 KB, corresponde a 4 operaciones de P1 (1 KB/bloque) y 8 de P2 (512 B/bloque). El funcionamiento lógico sería que cada vez que P1 acaba una operación se arranque la de P2, puesto que pueden funcionar en paralelo holgadamente. Por tanto, el tiempo de dos operaciones de P2 se añadiría al de cuatro de P1:

$$t_{operación4KB:P1-->P2} = 4 \cdot t_{operaciónP1} + 2 \cdot t_{operaciónP2} = 20,214556\ ms$$

También se podría considerar que las operaciones se secuencializan, y quedaría, por tanto, un tiempo total correspondiente a cuatro operaciones de P1 más ocho operaciones de P2.

1 *[6 puntos] Sea un procesador con tamaño de palabra de 64 bits, direccionamiento a nivel de byte y un espacio de direcciones físico de 4TB. Su sistema de memoria consta de una memoria principal con tiempo de acceso de 40 ns y un único nivel de memoria caché, compuesto por una caché de instrucciones (McaI) y otra de datos (McaD), con las siguientes características:*

- *Capacidad 32KB, bloques de 32B y tiempo de acceso de 2 ns.*
- *Ubicación asociativa por conjuntos de 2 bloques, política de lectura out of order fetch y política de reemplazo LRU.*
- *La política de escritura utilizada en la caché de datos es aplazada con actualización (CBWA). En los fallos de escritura, primero se realiza la escritura en Mp y a continuación se lleva el bloque a la caché.*
- *El tiempo empleado para leer o escribir una palabra en Mp es 40 ns, mientras que el tiempo en transferir un bloque entre Mp y Mca es 55 ns.*

En este procesador se ejecuta el siguiente fragmento de código.

```
for (i=0; i<32; i++) {
   vS[i] = 0;
   for (j=0; j<32; j++)
        vS[i] = vS[i] + (mA[i][j] * mB[i][j]);
}
```

Sabiendo que las variables i y j están asignadas a registros del procesador, los elementos del vector y de las matrices ocupan una palabra y están almacenados en Mp a partir de las siguientes direcciones expresadas en hexadecimal: vS: H'0 .. 02000, mA: H'0 .. 04000 y mB: H'0 .. 08000

a) *Especifique cómo interpreta la McaD las direcciones físicas (campos en los que se descompone y su longitud).*

b) *Calcule el número de bloques que ocupan las matrices y el vector. Determine asimismo los conjuntos de caché en que se alojarán.*

c) *Indique qué tipos de fallos se producirán en la McaD en la ejecución completa del código, sabiendo que inicialmente está invalidada. Determine si es posible que se reemplace algún bloque modificado.*

d) *Calcule el número de accesos y fallos que se producen en la McaD, distinguiendo entre lectura y escritura, y la tasa de aciertos que se obtiene.*

e) *Calcule el tiempo total empleado en los accesos a memoria, y el tiempo medio de acceso.*

f) *Suponga ahora que el código anterior se optimiza, sustituyéndolo por el que se muestra a continuación, donde r se asigna a un registro del procesador:*

```
for (i=0; i<32; i++) {
   r = 0;
   for (j=0; j<32; j++)
        r = r + (mA[i][j] * mB[i][j]);
   vS[i] = r;
}
```

f.1) *Calcule la nueva tasa de aciertos de la McaD, así como el tiempo total empleado en los accesos a memoria y el tiempo medio.*

f.2) *Calcule la ganancia o speedup que se consigue con esta versión frente a la original.*

SOLUCIÓN

a) Las direcciones físicas constan de 42 bits, para direccionar los 4TB. La McaD tiene capacidad para $32KB/32B/bloque = 2^{10}$ *bloques*, y al ser su política de ubicación asociativa por conjuntos de 2 bloques, está compuesta por: 2^{10} *bloques*$/2$ *bloques*$/conjunto = 2^9 = 512$ *conjuntos*

Por lo tanto, la McaD interpreta las direcciones físicas de la siguiente forma:

etiqueta	conjunto	byte en bloque
28 bits	9 bits	5 bits

b) Cada matriz ocupa (32 *filas* $\times$ 32 *columnas* $\times$ $8B/elemento$) $= 2^{13}B$, y cada bloque tiene capacidad para 32B, por lo que cada matriz ocupa $2^{13}B/2^5B/bloque = 256$ *bloques* de Mp.

El vector consta de 32 *elementos* $\times$ $8B/elemento = 2^8B \rightarrow$ $2^8B/2^5B/bloque = 8$ *bloques* de Mp.

En total, el número de bloques pedido es: $2 \times 256 + 8 = 520$ *bloques*, menor que la capacidad de la McaD (1.024 bloques).

Teniendo en cuenta las direcciones de comienzo del vector y las matrices, se puede comprobar que éstas están alineadas a bloque (sus 5 bits menos significativos tienen valor cero), por lo que los siguientes 9 bits indican el conjunto en que se alojará el primer bloque de cada uno:

vS: H'0 ... 02000 $\rightarrow$ conjunto = B'100000000 $\rightarrow$ 256

mA: H'0 ... 04000 $\rightarrow$ conjunto = B'000000000 $\rightarrow$ 0

mB: H'0 ... 08000 $\rightarrow$ conjunto = B'000000000 $\rightarrow$ 0

En consecuencia, los 8 bloques que ocupa vS se alojarán en uno de los bloques de los conjuntos 256 a 263, mientras que los 256 bloques que ocupan mA y mB se alojarán en los conjuntos 0 a 255, ocupando entre ambas los dos bloques de dichos conjuntos.

c) Los únicos fallos que se pueden producir son de primera referencia ya que la caché tiene capacidad suficiente y no se pueden producir fallos por conflicto, de acuerdo a lo expuesto en el apartado anterior. Por lo tanto, tampoco se pueden producir reemplazos de bloques modificados.

d) Como se hace referencia a un total de 520 bloques distintos, se producirán 520 fallos de primera referencia, en el acceso a la primera palabra de cada bloque, de los que 8 son de escritura (vS[i]=0) y el resto de lectura (mA y mB).

El número total de accesos es:

$32 + 32 \times 32$ *escrituras* $+ 32 \times 32 \times 3$ *lecturas* $= 1.056$ *escrituras* $+ 3.072$ *lecturas* $= 4.128$ *accesos*

y la tasa de aciertos: $((4.128$ *accesos* $- 520$ *fallos*$)/4.128$ *accesos*$) \times 100 = 87,4\,\%$

e) El tiempo total empleado en los accesos a datos se calcula como:

$Ttotal = N°aciertos \times Tacierto + N°fallos \times Tfallo = (4.128 - 520) \times 2 + 520 \times (2 + 40) = 29.056$ *ns*

Nótese que el tiempo de acceso cuando se produce un fallo ($Tfallo$) es el mismo independientemente de que el fallo sea de lectura o de escritura, debido a la política de lectura utilizada (*out of order fetch*) y al modo de actuación cuando se produce un fallo en escritura (primero se escribe en Mp y a continuación se lleva el bloque a caché.

Para calcular el tiempo medio de acceso basta con dividir el tiempo anterior por el número total de accesos:

$\overline{T} = 29.056/4.128 = 7$ *ns*

f) En el código optimizado los fallos son los mismos: 8 de escritura (vS[i] = r) y 512 de lectura (mA y mB). El número total de accesos disminuye, ya que en este caso se producen:

$32 \times 32 \times 2$ *lecturas* $+ 32 \times 1$ *escrituras* $= 2.048$ *lecturas* $+ 32$ *escrituras* $= 2.080$ *accesos*

$Hr(McaD) = ((2.080$ *accesos* $- 520$ *fallos*$)/2.080$ *accesos*$) \times 100 = 75\,\%$

$Ttotal = (2.080 - 520) \times 2 + 520 \times (2 + 40) = 24.960$ *ns*

$\overline{T} = 24.960/2.080 = 12$ *ns*

$Ganancia = 29.056/24.960 = 1,16$

2 *[4 puntos] Considere ahora que el procesador del problema anterior emplea memoria virtual paginada con las siguientes características:*

- *Páginas de 4KB y cuatro niveles de tablas de páginas. Cada entrada de cada tabla de páginas ocupa 1 palabra.*
- *TLBs separadas para instrucciones y datos, cada una con 16 entradas, totalmente asociativas, y tiempo de acceso 1 ns.*

a) Indique razonadamente cuántas entradas de TLB de datos se utilizarán en la ejecución del código anterior.

b) Indique razonadamente si es posible solapar el acceso a la MaD con el acceso a la TLB.

c) Calcule los tiempos mínimo y máximo de acceso y de ocupación al sistema de memoria, en la ejecución del código anterior, indicando claramente en qué condiciones se produce cada uno. Suponga que no se produce ningún fallo de página y considere accesos tanto de lectura como de escritura.

d) Si en lugar de tener 4 niveles de TP hubiese un solo nivel ¿en cuánto se reduciría el tiempo máximo de acceso a memoria? ¿Qué ventaja proporciona entonces tener 4 niveles de tablas de páginas?

SOLUCIÓN

a) Ya que los 12 bits menos significativos de las direcciones de comienzo del vector y las matrices tienen valor cero, esto significa que están alineadas a página. Teniendo en cuenta el tamaño de las páginas y de los datos, el número de páginas que ocupan los datos es:

$(32 \times 32 \times 8B)/4KB/pagina = 2^{13}/2^{12} = 2$ páginas cada matriz

$(32 \times 8B)/4KB/pagina = 2^{8}/2^{12} = 1$ páginas para el vector

En consecuencia, de las 16 entradas de que dispone la TLB se utilizarán únicamente cinco.

b) No se puede solapar el acceso a ambos dispositivos, ya que las páginas son de 4KB ($2^{12}B$) y con los 12 bits que no se traducen no se pueden direccionar los 2^9 conjuntos, más el byte dentro del bloque ($2^5B/bloque$).

c) El tiempo mínimo, tanto de acceso como de ocupación, se producirá en caso de acierto en TLB y en caché.

$T_{min} = T_{TLB} + T_{Mca} = 1 + 2 = 3\ ns$

El tiempo máximo de acceso corresponde a un acceso con fallo de traducción en la TLB, con la correspondiente consulta a los cuatro niveles de tablas residentes en Mp. Una vez realizada la traducción, el tiempo máximo de acceso a la información corresponde a un acceso con fallo en caché, no habiendo que considerar el caso en el que el bloque a reemplazar ha sido modificado.

$T_{max(Acc)} = T_{TLB} + T_{TP's} + T_{Mca} + T_{Mp} = 1 + 4 \times 40 + 2 + 40 = 203\ ns$

Obsérvese que no se ha tenido en cuenta el caso en el que el bloque a reemplazar ha sido modificado, pues según se expuso en el ejercicio anterior no se puede dar en el código del enunciado.

Obsérvese también que el tiempo máximo calculado es el mismo independientemente de que el acceso sea de lectura o de escritura, ya que la política de lectura es *out of order fetch* y cuando se produce un fallo de escritura primero se escribe en Mp y a continuación se envía a Mca el bloque que produce el fallo.

El tiempo máximo de ocupación depende del tipo de acceso:

- Lectura:

  $T_{max(Ocup)} = T_{TLB} + T_{TP's} + T_{Mca} + T_{blqMp \rightarrow Mca} = 1 + 4 \times 40 + 2 + 55 = 218\ ns$
- Escritura:

  $T_{max(Ocup)} = T_{TLB} + T_{TP's} + T_{Mca} + T_{Mp} + T_{blqMp \rightarrow Mca} = 1 + 4 \times 40 + 2 + 40 + 55 = 258\ ns$

Obsérvese, que en este caso el tiempo de ocupación es mayor en las escrituras, ya que cuando se produce un fallo primero se escribe en Mp y a continuación se actualiza la caché.

d) Si hubiera una única tabla de páginas sería necesario un único acceso a Mp en caso de fallo de TLB, por lo que el tiempo máximo de acceso se reduciría en $3 \times 40 = 120\ ns$, quedando: $T_{max(Acc)} = 203\ ns - 120\ ns = 83\ ns$

1 *En un computador cuyo procesador dispone de un pipeline de 5 etapas se ejecuta el programa que se muestra a continuación, desarrollado inicialmente para un procesador sin pipeline.*

```
(1)         add r10, r0, r0     ; i=0                    |
(2)  BUCI:  add r11, r0, r0     ; j=0                    |for (i=0; i<M; i++)
(3)  BUCJ:  ld r20, #0(r15)     ; r20 <- mA(i,j)         |  for (j=0; j<N; j++)
(4)         ld r21, #0(r16)     ; r21 <- mB(i,j)         |    if (mA[i][j] < mB[i][j])
(5)         sub r2, r20, r21    ; r2 <- r20-r21          |      mA[i][j] = mB[i][j];
(6)         bge r2, $SIGJ       ; si r20>r21 ir a SIGJ   |
(7)         st r21, 0(r15)      ; mA(i,j)=r3             |
(8)  SIGJ:  add r15, r15, #4    ;                        |
(9)         add r16, r16, #4    ;                        |
(10)        add r11, r11, #1    ; j++                    |
(11)        cmp r2, r11, N      ;                        |
(12)        blt r2, $BUCJ       ; si j<N ir a BUCJ       |
(13)        add r10, r10, #1    ; i++                    |
(14)        cmp r2, r10, M      ;                        |
(15)        blt r2, $BUCI       ; si i<M ir a BUCI       |
(16)        add r25, r0, r0     ; k=0                    |k = 0;
```

Se desea analizar el comportamiento de este programa al ejecutarlo en dos procesadores semejantes, con el mismo pipeline pero con diferente tratamiento de los saltos. En ambos casos el pipeline del procesador dispone de todo tipo de mecanismos de adelantamiento y emplea las siguientes etapas:

- *E1: Fetch e incremento del PC.*
- *E2: Decodificación, lectura de registros, evaluación de la condición de salto.*
- *E3: Ejecución y cálculo de la dirección, tanto para load/store como para las instrucciones de salto.*
- *E4: Acceso a memoria para lectura o escritura de datos.*
- *E5: Escritura en registros.*

En cuanto a las diferencias en el tratamiento de los saltos, son las siguientes:

- ***ProcA**: Utiliza predicción de NO-SALTO.*
- ***ProcB**: Dispone de predicción dinámica, con dos bits inicializados a '01'(no salta, predicción débil). La predicción se realiza en la etapa 1: Fetch.*

Se pide:

***a) (1 punto)** Describa las dependencias* de datos *y los ciclos de parada que se gastan en cada una de ellas en cada uno de los procesadores.*

***b) (1 punto)** Indique cuántos ciclos de penalización introduce **ProcA** en cada salto (tomado/no tomado).*

***c) (2 puntos)** Calcule para el **ProcA** los ciclos totales consumidos al ejecutar el programa dado, con $M = N = 100$ y habiendo observado que en la bifurcación correspondiente a la instrucción I6 el 40 % de las ocasiones se toma (T) el salto y en el 60 % restante no se toma (N).*

***d) (1 punto)** Calcule el CPI correspondiente a la ejecución del programa dado en el **ProcA**.*

***e) (1,5 puntos)** Calcule cómo cambiaría el CPI si el procesador indicado utilizase salto retardado (*delay-slot*) y los huecos necesarios se hubieran rellenado con instrucciones* nop. *Indique también la ganancia o* speedup *obtenido, justificando si corresponde a una mejora o a una disminución de las prestaciones.*

***f) (1 punto)** Indique los ciclos de penalización que introduce el **ProcB** cada vez que acierta y cada vez que falla la predicción de salto.*

***g) (1,5 puntos)** Calcule los ciclos totales consumidos al ejecutar el programa dado, sabiendo que de nuevo $M = N = 100$ y, suponiendo además que el 50 % de las veces se acierta en la predicción de salto correspondiente a la instrucción I6.*

***h) (1 punto)** Determine cuál es el porcentaje real de aciertos y fallos en la predicción del salto correspondiente a la instrucción I6 si se ha observado al ejecutar el programa dado que el patrón de saltos T/N (Tomados/No tomados) es siempre el siguiente:*

(salto T/N): $T\ T\ N\ N\ N\ N\ T\ N\ N\ T$
$T\ T\ N\ N\ N\ N\ T\ N\ N\ T$
$.\ .\ .\ .\ .\ .\ .\ .\ .\ .$

SOLUCIÓN

a) Al disponer de mecanismos de adelantamiento, solo se introducen ciclos de parada por DD en los casos siguientes (un ciclo en cada caso):

I4 → I5, por $r21$; I5 → I6 por $r2$; I11 → I12, por $r2$; I14 → I15, por $r2$

b) No hay penalización en el caso de saltos no tomados. En los saltos que sí se toman, se introducen 2 ciclos de penalización ya que hasta la etapa E3 no se conoce la dirección destino del salto.

c) Ciclos debidos a ejecución de instrucciones:

Instrucciones: $1 + 100 \times (1 + 100 \times (0,4 \times 9 + 0,6 \times 10) + 3) + 1 = 96.402$ ciclos

De las DD descritas en el primer apartado, 3 se producen en el bucle interno (j) y 1 en el bucle externo (i):

Penalización DD: $3 \times 100 \times 100 + 1 \times 100 = 30.100$ ciclos

Las DC se producen en las instrucciones I6, I12 y I15. En el caso de la I6, el 60 % de los saltos no se toman (no penalizan). La I12 pertenece al bucle interno, que se ejecuta 100 veces por cada vez que se ejecuta el bucle externo. De esas 100, en 99 ocasiones se toma el salto y solo en la última no se toma. La I15 pertenece al bucle externo, que se ejecuta 100 veces, de las que las 99 primeras se toma el salto.

Penalización DC: $(100 \times 100 \times (0,4 \times 2)) + (100 \times 99 \times 2) + (99 \times 2) = 8.000 + 19.800 + 198 = 27.998$ ciclos

$\mathbf{T_{Ejecución}} = 96.402 + 30.100 + 27.998 = \mathbf{154.500\ ciclos}$

d) El CPI será simplemente:

$\mathbf{CPI} = 154.500/96.402 = \mathbf{1,6\ ciclos/instrucción}$

e) No se modifica el número de instrucciones *útiles* ejecutadas ni la penalización por DD. Se requieren 2 instrucciones `NOP` tras cada salto, por lo que, independientemente de que se tome o no se tome:

Penalización DC: $(100 \times 100 \times 2)) + (100 \times 100 \times 2) + (100 \times 2) = 20.000 + 20.000 + 200 = 40.200$ ciclos

$\mathbf{T_{Ejecución}} = 96.402 + 30.100 + 40.200 = \mathbf{166.702\ ciclos}$

$\mathbf{CPI} = 166.702/96.402 = \mathbf{1,73\ ciclos/instrucción}$

$\mathbf{Ganancia(speedup)} = 154.500/166.702 = \mathbf{0,927\ (más\ lento)}$

f) Se utiliza predicción de salto en la etapa E1:fetch. Un acierto en la predicción supone que no se penaliza con ningún ciclo la instrucción de salto. Un fallo en la predicción consume dos ciclos de penalización, hasta que se obtiene la dirección destino del salto.

g) El número de ciclos de penalización ahora depende de que se acierte o falle en la predicción.

En la instrucción I6, se produce un fallo de predicción el 50 % de las veces. La instrucción I12 pertenece al bucle interno y falla la predicción la primera vez que se pasa por él y siempre que llega a la última iteración (otras 100 veces). La instrucción I15 pertenece al bucle externo, que se ejecuta 100 veces de las que las 99 primeras se toma el salto. Falla la predicción en la primera iteración y en la última.

Penalización DC: $(100 \times 100 \times (0,5 \times 2)) + ((1 + 100) \times 2) + (2 \times 2) = 10.000 + 202 + 4 = 10.206$ ciclos

$\mathbf{T_{Ejecución}} = 96.402 + 30.100 + 10.206 = \mathbf{136.708\ ciclos}$

h) Se producirán los siguientes cambios en los bits de predicción y en el resultado de la predicción (Acierto/Fallo):

Salto	T	T	N	N	N	N	T	N	N	T
B0	0	1	1	1	0	0	0	0	0	0
B1	1	0	1	0	1	0	0	1	0	0
PRED	F	A	F	F	A	A	F	A	A	F

Como la secuencia anterior se repite, se produce el **50 %** de aciertos de predicción, que coincide con el supuesto en el apartado anterior.

2 Sea un multiprocesador de memoria compartida con 16 procesadores, conectados a través de un bus de ciclo partido a la memoria principal. Cada procesador dispone de una memoria cache privada de datos, que por simplificar se va a considerar que tiene sólo cuatro bloques, es de ubicación directa y que cada bloque tiene sólo dos palabras.

Para mantener la coherencia usa la política de invalidación MESI, con cuatro estados, sobre una implementación *snoopy*

Indique las acciones que desencadenan las siguientes operaciones y qué cambios se producen tanto en las cachés como en la memoria principal. Suponga que cada operación es independiente de las demás, por lo que se parte del estado inicial mostrado en la siguiente figura. Para simplificar la figura, el tag contiene la dirección completa.

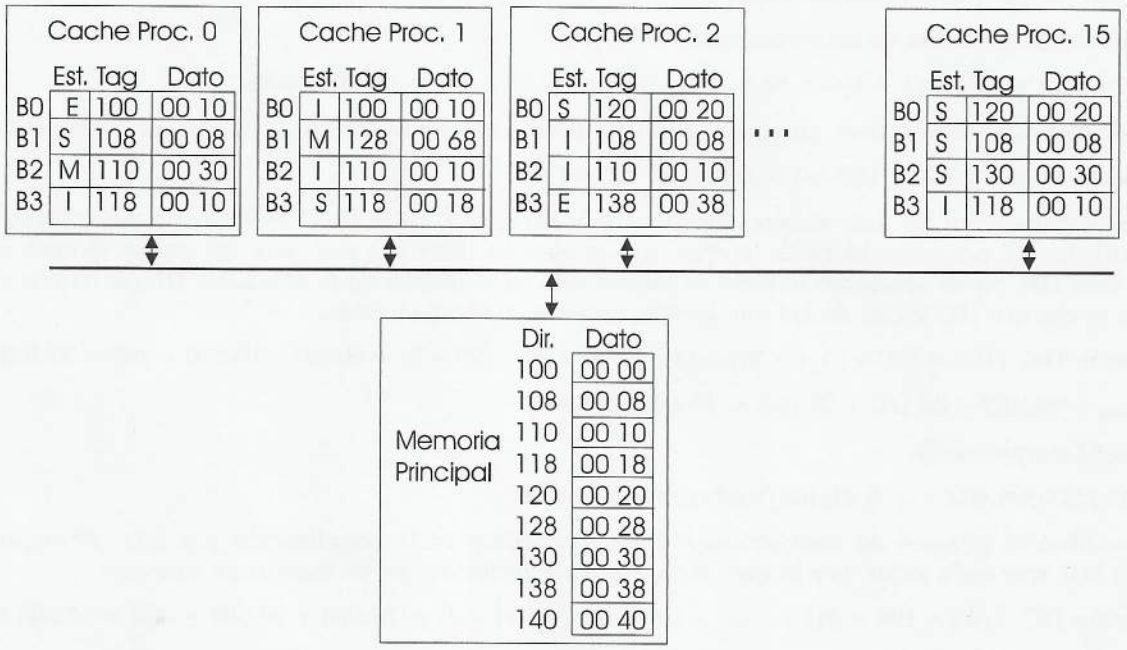

Figura 1 Multiprocesador con coherencia de caches basada en MESI

a) Procesador 15 lee el dato de la dirección 118.

b) Procesador 1 escribe 80 en la dirección 128.

c) Procesador 2 escribe 80 en la dirección 130.

d) Procesador 2 lee el dato de la dirección 128.

e) Procesador 1 lee el dato de la dirección 138.

f) Procesador 1 escribe 80 en la dirección 108.

SOLUCIÓN

a) Procesador 15 lee el dato de la dirección 118. Da un fallo y genera una petición de lectura en el bus, BusRd, dirección 118. El contenido de la cache será P15.B3 (S 118 00 18)

b) Procesador 1 escribe el dato de la dirección 128. Da un acierto en cache y no genera ninguna petición en el bus. El contenido de la cache será P1.B1 (M 128 00 80)

c) Procesador 2 escribe 80 en la dirección 130. Genera una petición en el bus de lectura con invalidación, BusRdX, dirección 130. Al final el contenido de las caches cambia en: P2.B2 (M 130 00 80) y P15.B2 (I 130 00 30)

d) Procesador 2 lee el dato de la dirección 128. Genera una petición de lectura en el bus, BusRd, dirección 128. Como está en P1 en estado modificado debe actualizarse la memoria. Al final el contenido de las caches cambia en: P1.B1 (S 128 00 68) P2.B1 (S 128 00 68) y la memoria M[128] = 00 68

e) Procesador 1 lee el dato de la dirección 138. Genera una petición de lectura en el bus, BusRd, dirección 138. Al final el contenido de las caches cambia en: P1.B3 (S 138 00 38) P2.B3 (S 138 00 38)

f) Procesador 1 escribe 80 en la dirección 108. Genera una petición en el bus de lectura con invalidación, BusRdX, dirección 108. Como el dato del bloque 1 está modificado debe actualizarse la memoria principal. Al final el contenido de las caches cambia en: P1.B1 (M 108 00 80), P0.B1 (I 108 00 08), P15.B1 (I 108 00 08) y la memoria M[128] = 00 68

3 Sea una memoria entrelazada de 8 módulos, con entrelazado de orden inferior, entrelazado complejo, con palabra de 64 bits y tiempo de acceso de 40 ns.

Dados los siguientes fragmentos de código, determine el ancho de banda que se obtiene en cada caso.

```
a. for (i=0; i < 1000; i+=1)
       a = a + v[i];
b. for (i=0; i < 1000; i+=3)
       a = a + v[i];
c. for (i=0; i < 1000; i+=4)
       a = a + v[i];
d. for (i=0; i < 1000; i+=5)
       a = a + v[i];
```

SOLUCIÓN

Como el tiempo de acceso es de 40 ns, realizará un máximo de 25 millones de accesos por segundo, cada uno de los cuales proporciona 64 bits, es decir, 8 bytes. Por tanto cada módulo tiene un ancho de banda de $25 \cdot 10^6 \times 8bytes = 200MB/s$.

Al usar entrelazado complejo, se puede usar una dirección distinta para cada módulo:

- Código a. Direcciones: 0, 1, 2, 3, 4, 5, 6, 7, 8... Módulos: 0, 1, 2, 3, 4, 5, 6, 7, 0... por lo que el ancho de banda es $8 \times 200MB/s = 1600MB/s$.
- Código b. Direcciones: 0, 3, 6, 9, 12, 15, 18, 21, 24... Módulos: 0, 3, 6, 1, 4, 7, 2, 5, 0... por lo que no hay colisiones y el ancho de banda es $8 \times 200MB/s = 1600MB/s$.
- Código c. Direcciones: 0, 4, 8, 12, 16, 20, 24, 28,... Módulos: 0, 4, 0, 4, 0, 4, 0, 4,... por lo que el ancho de banda es $2 \times 200MB/s = 400MB/s$.
- Código d. Direcciones: 0, 5, 10, 15, 20, 25, 30, 35, 40... Módulos: 0, 5, 2, 7, 4, 1, 6, 3, 0... por lo que no hay colisiones y el ancho de banda es $8 \times 200MB/s = 1600MB/s$.